**LAPORAN PRAKTIKUM JAVASCRIPT**

“Aplikasi To-Do List”

****

**Dibuat Oleh:**

Nama : Handra Putra Alma

NPM : 4523210052

Mata Kuliah : DESAIN WEB – A

Tanggal : 01/10/2024

**Dosen Pengampu:**

Adi Wahyu Pribadi,  S.Si., M.Kom

**PRODI TEKNIK INFORMATIKA**

**UNIVERSITAS PANCASILA**

**JAKARTA**

**2024**

**BAB I**

PENDAHULUAN

* 1. **Tujuan**

Dengan menambahkan tugas baru, menandai tugas sebagai selesai, dan menghapus tugas yang tidak relevan, aplikasi To-Do List dirancang untuk membantu pengguna mengelola tugas sehari-hari. Aplikasi ini juga memiliki fitur untuk memfilter tugas berdasarkan statusnya (semua, aktif, atau selesai).

* 1. **Deskripsi**

Aplikasi to-do list adalah alat digital yang membantu pengguna mencatat, mengelola, dan melacak pekerjaan yang harus dilakukan. Untuk meningkatkan produktivitas, aplikasi ini memiliki fitur yang memungkinkan pengaturan prioritas, pengingat, dan pencapaian target harian atau jangka panjang.

* 1. **Penjelasan**

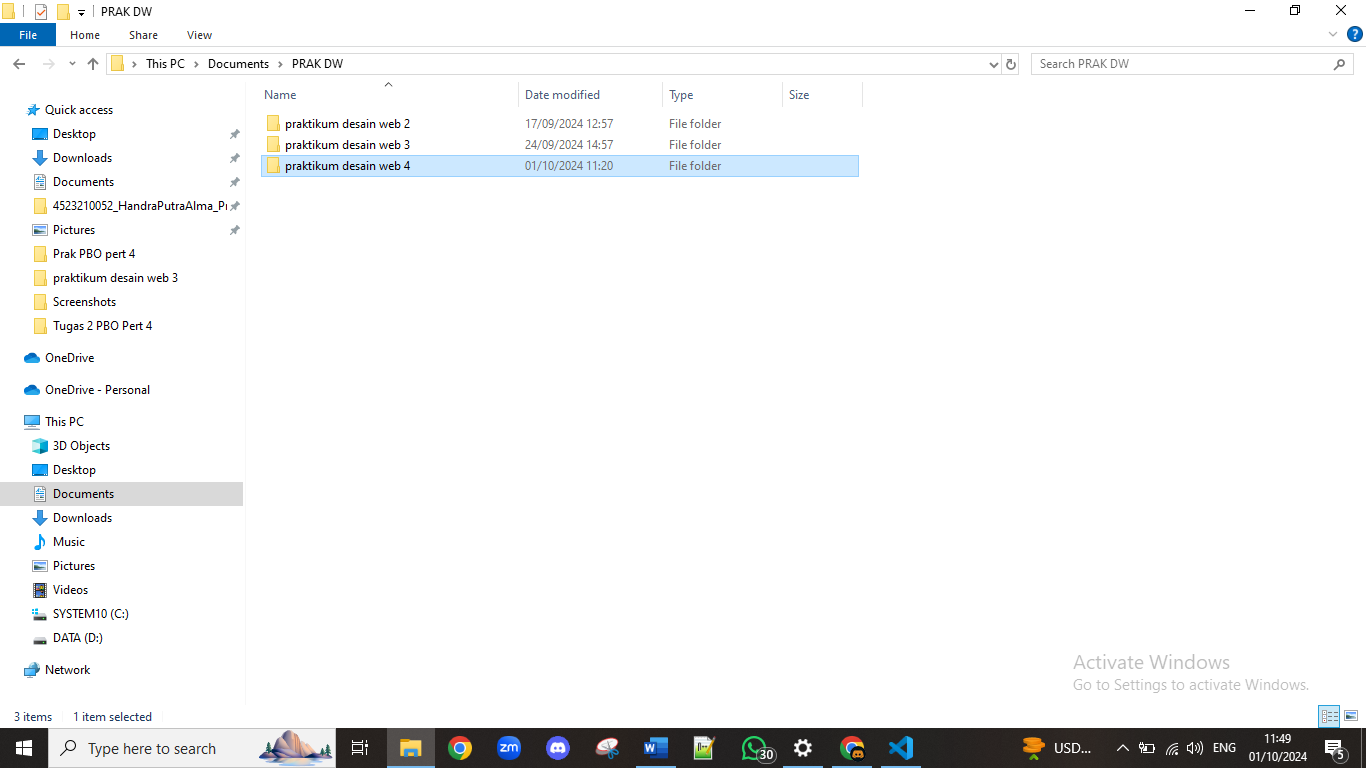
JavaScript adalah bahasa pemrograman yang digunakan untuk membuat halaman web lebih dinamis, interaktif, dan responsif. Itu bekerja di sisi browser, memungkinkan pengembang mengubah konten HTML/CSS, menangani peristiwa pengguna, membuat animasi, dan berinteraksi dengan server melalui AJAX atau Fetch API.

**BAB II**

LANGKAH PENGERJAAN

**2.1 Pengerjaan**

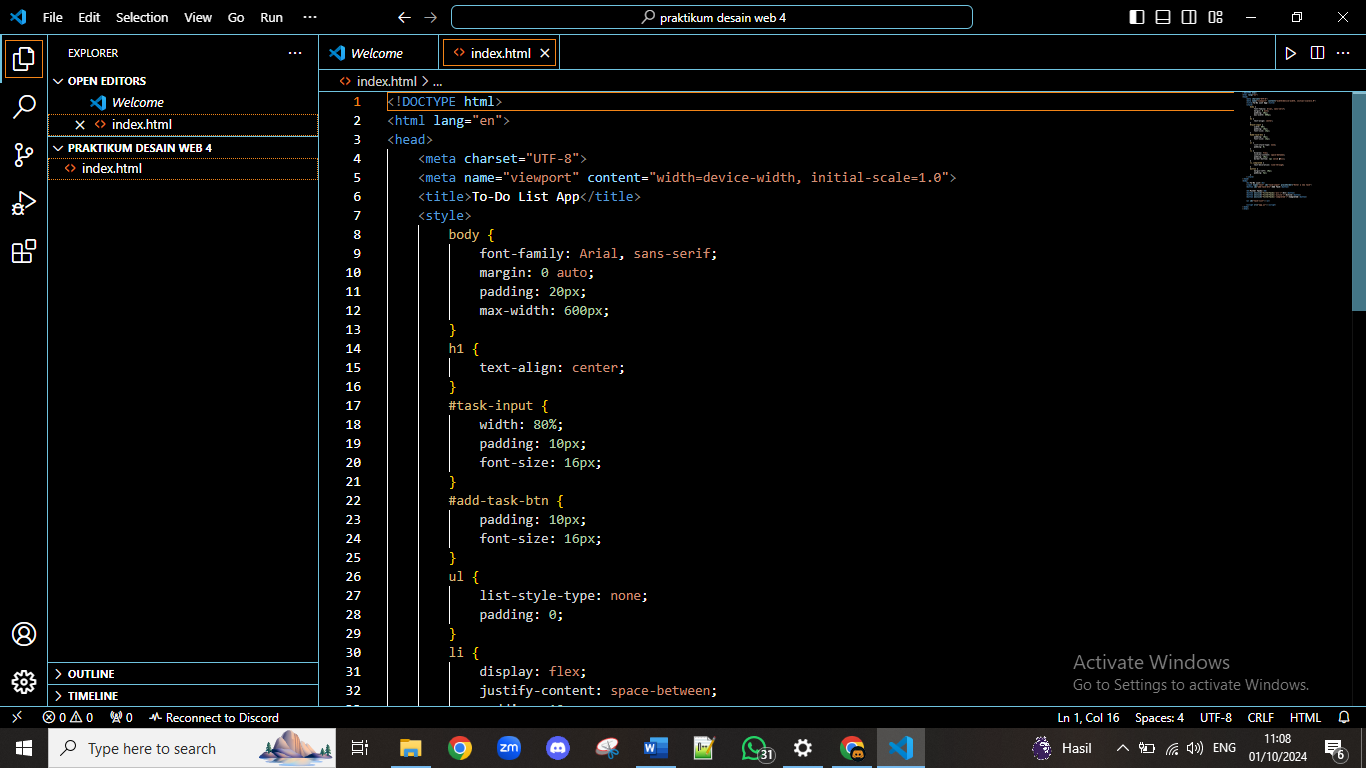
1. Menyiapkan folder pengerjaan

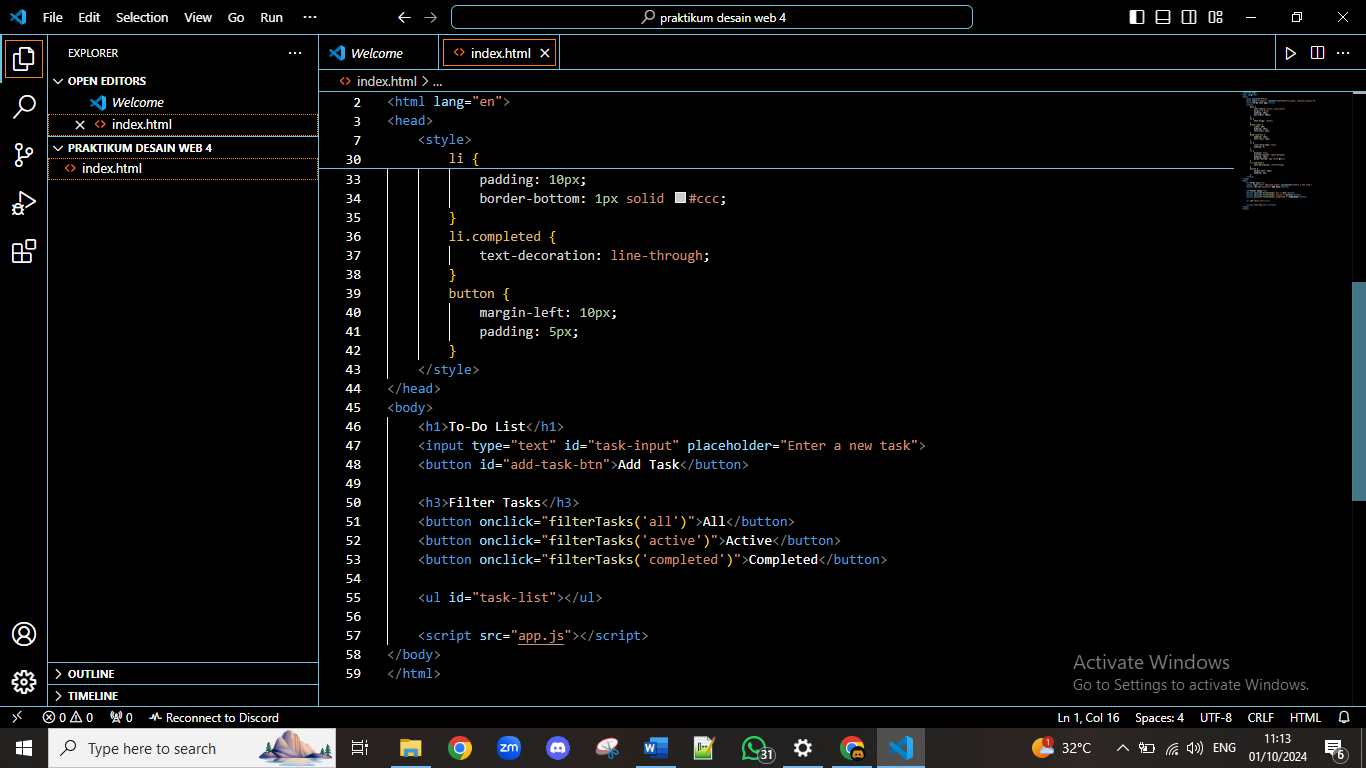


1. Menyiapkan File HTML, CSS & JavaScript



1. Membuat halaman HTML & CSS

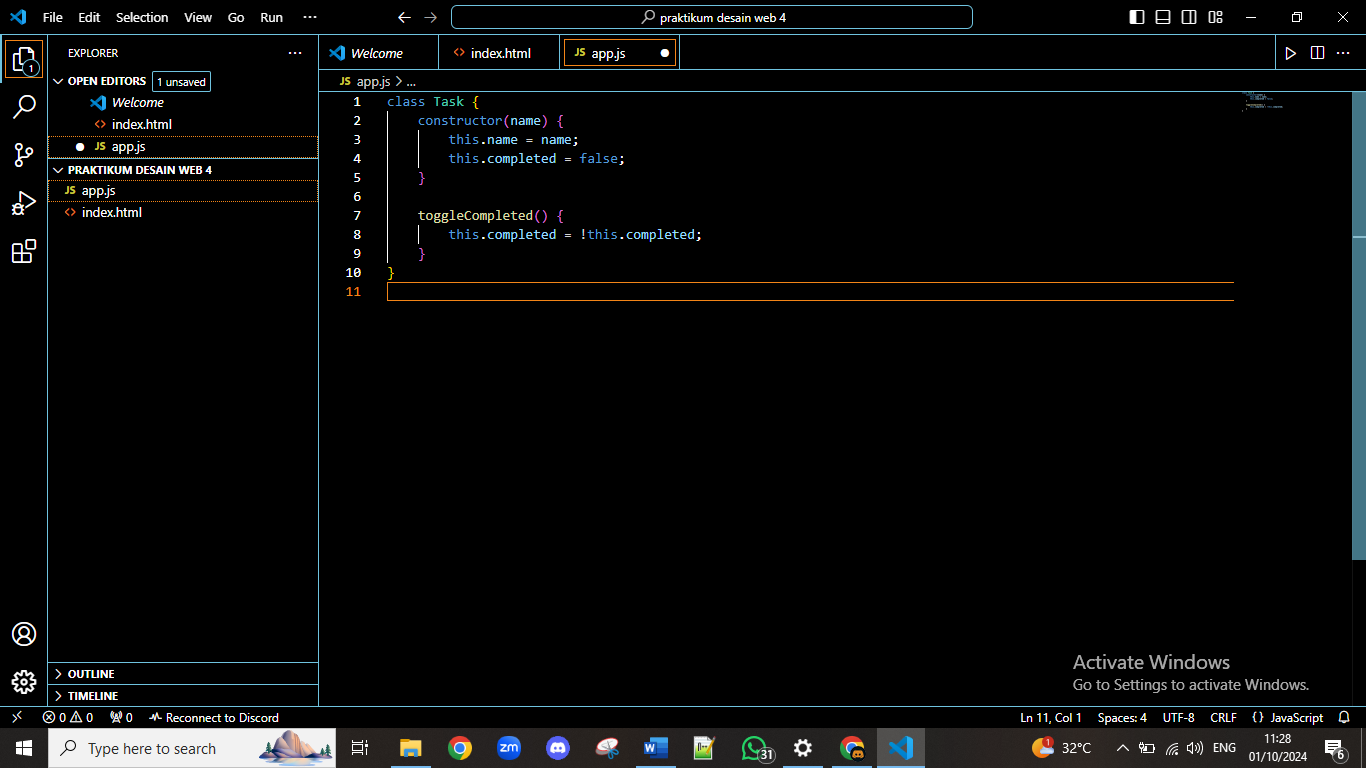




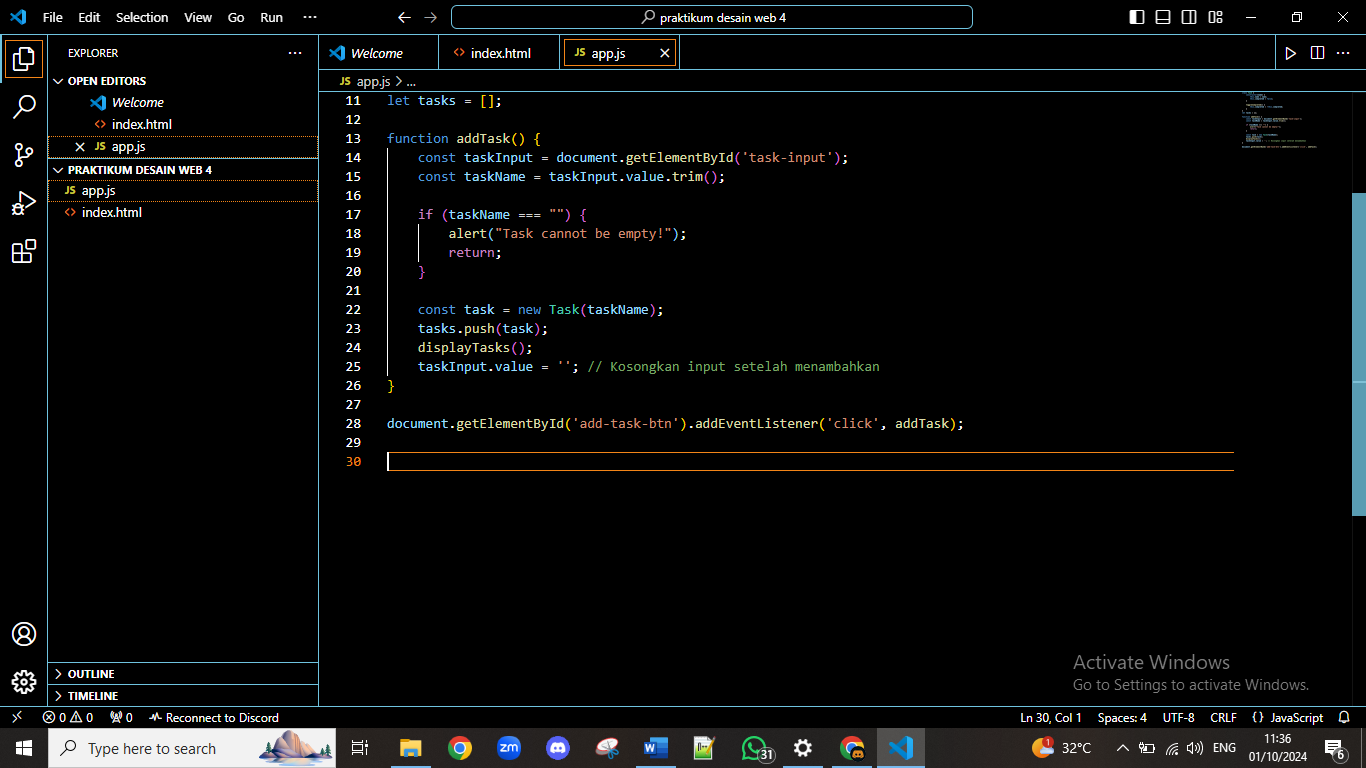
1. Membuat JavaScript

Class & Object Untuk Tugas

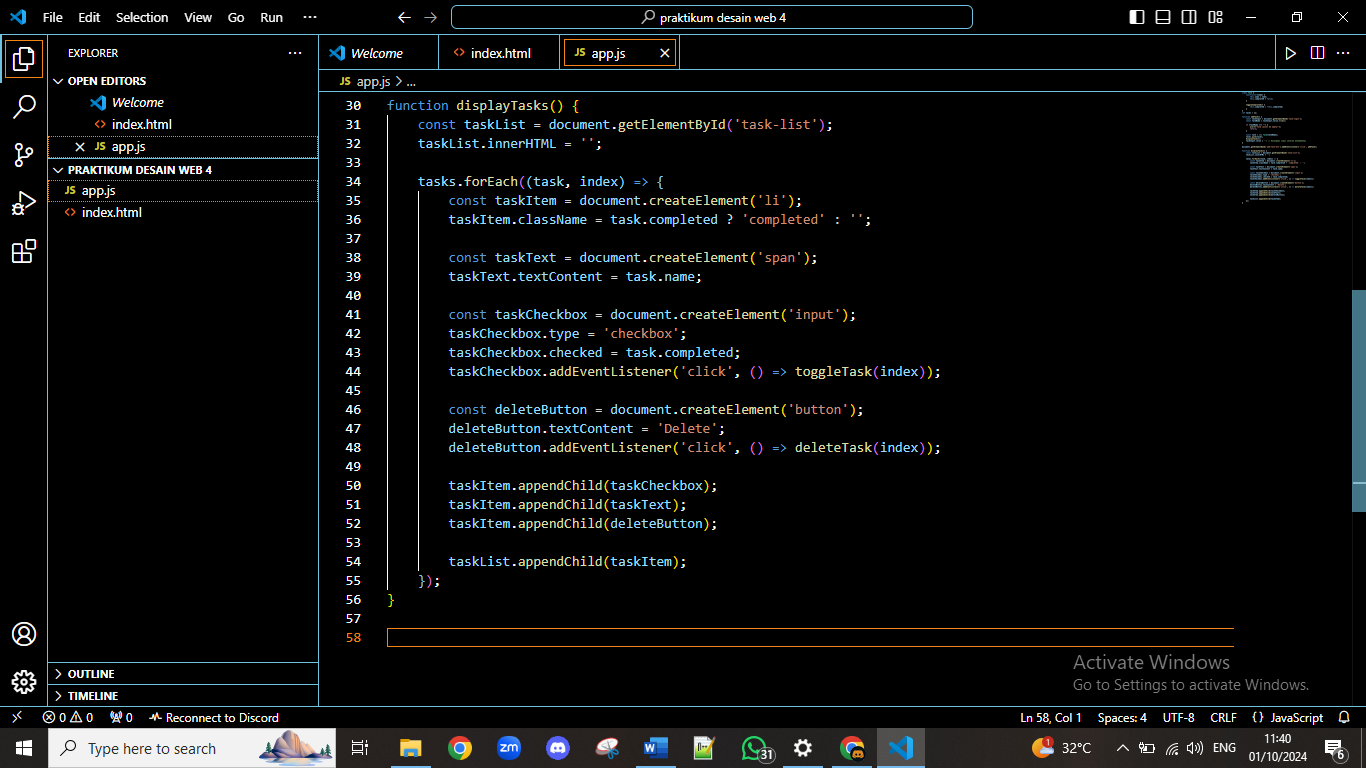
app.js → class untuk merepresentasikan tugas yang memiliki properti name dan completed



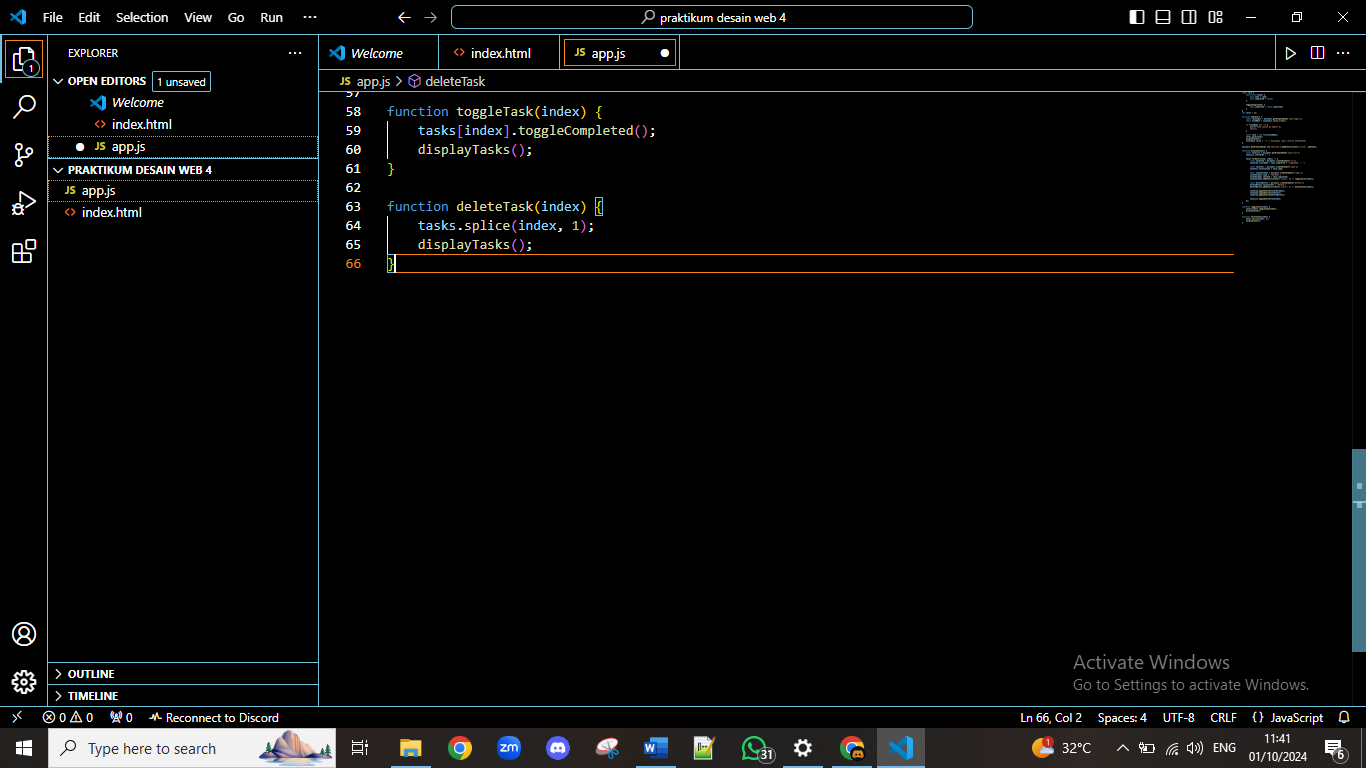
1. Membuat Fungsi untuk Menambah Tugas



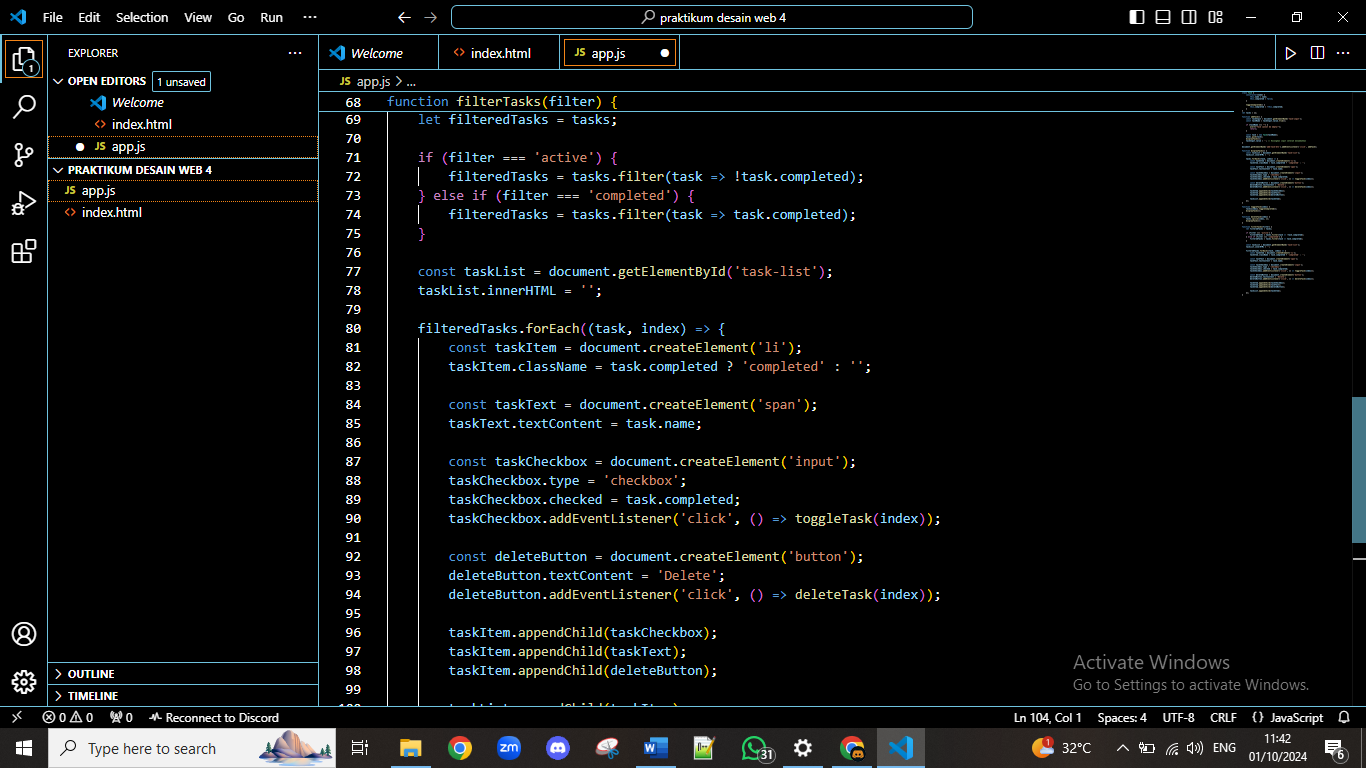
1. Menampilkan Tugas di DOM



1. Menandai Tugas Selesai dan Menghapus Tugas



1. Filter Tugas Berdasarkan Status



**BAB III**

Struktur dan Penjelasan Kode

**3.1 Source code file app.js**

class Task {

    constructor(name) {

        this.name = name;

        this.completed = false;

    }

    toggleCompleted() {

        this.completed = !this.completed;

    }

}

let tasks = [];

function addTask() {

    const taskInput = document.getElementById('task-input');

    const taskName = taskInput.value.trim();

    if (taskName === "") {

        alert("Task cannot be empty!");

        return;

    }

    const task = new Task(taskName);

    tasks.push(task);

    displayTasks();

    taskInput.value = ''; // Kosongkan input setelah menambahkan

}

document.getElementById('add-task-btn').addEventListener('click', addTask);

function displayTasks() {

    const taskList = document.getElementById('task-list');

    taskList.innerHTML = '';

    tasks.forEach((task, index) => {

        const taskItem = document.createElement('li');

        taskItem.className = task.completed ? 'completed' : '';

        const taskText = document.createElement('span');

        taskText.textContent = task.name;

        const taskCheckbox = document.createElement('input');

        taskCheckbox.type = 'checkbox';

        taskCheckbox.checked = task.completed;

        taskCheckbox.addEventListener('click', () => toggleTask(index));

        const deleteButton = document.createElement('button');

        deleteButton.textContent = 'Delete';

        deleteButton.addEventListener('click', () => deleteTask(index));

        taskItem.appendChild(taskCheckbox);

        taskItem.appendChild(taskText);

        taskItem.appendChild(deleteButton);

        taskList.appendChild(taskItem);

    });

}

function toggleTask(index) {

    tasks[index].toggleCompleted();

    displayTasks();

}

function deleteTask(index) {

    tasks.splice(index, 1);

    displayTasks();

}

function filterTasks(filter) {

    let filteredTasks = tasks;

    if (filter === 'active') {

        filteredTasks = tasks.filter(task => !task.completed);

    } else if (filter === 'completed') {

        filteredTasks = tasks.filter(task => task.completed);

    }

    const taskList = document.getElementById('task-list');

    taskList.innerHTML = '';

    filteredTasks.forEach((task, index) => {

        const taskItem = document.createElement('li');

        taskItem.className = task.completed ? 'completed' : '';

        const taskText = document.createElement('span');

        taskText.textContent = task.name;

        const taskCheckbox = document.createElement('input');

        taskCheckbox.type = 'checkbox';

        taskCheckbox.checked = task.completed;

        taskCheckbox.addEventListener('click', () => toggleTask(index));

        const deleteButton = document.createElement('button');

        deleteButton.textContent = 'Delete';

        deleteButton.addEventListener('click', () => deleteTask(index));

        taskItem.appendChild(taskCheckbox);

        taskItem.appendChild(taskText);

        taskItem.appendChild(deleteButton);

        taskList.appendChild(taskItem);

    });

}

**3.2 Penjelasan bagian dari kode**

**Class Task dan fungsinya.**

* Class Task merepresentasikan sebuah tugas/to-do. Setiap instance dari Task memiliki properti name (nama tugas) dan completed (status apakah tugas sudah selesai atau belum).
* Fungsi toggleCompleted() digunakan untuk membalik status completed dari false menjadi true, atau sebaliknya.

**Fungsi addTask, toggleTask, deleteTask, dan filterTasks.**

* **addTask()**: Fungsi ini digunakan untuk menambahkan tugas baru ke dalam daftar tugas. Mengambil input dari elemen HTML (input teks), membuat objek Task, menambahkannya ke array tasks, dan memanggil displayTasks() untuk menampilkan tugas yang baru.
* **toggleTask()**: Mengubah status selesai dari tugas. Diterapkan ketika checkbox di klik, dan memanggil metode toggleCompleted() untuk tugas yang dipilih.
* **deleteTask()**: Menghapus tugas berdasarkan indeks dari array tasks. Setelah tugas dihapus, fungsi displayTasks() dipanggil untuk memperbarui tampilan daftar.
* **filterTasks()**: Menyaring tugas berdasarkan status completed. Jika filter yang dipilih adalah 'active', hanya menampilkan tugas yang belum selesai, dan jika 'completed', hanya menampilkan tugas yang sudah selesai.

**Cara penggunaan manipulasi DOM untuk menampilkan tugas.**

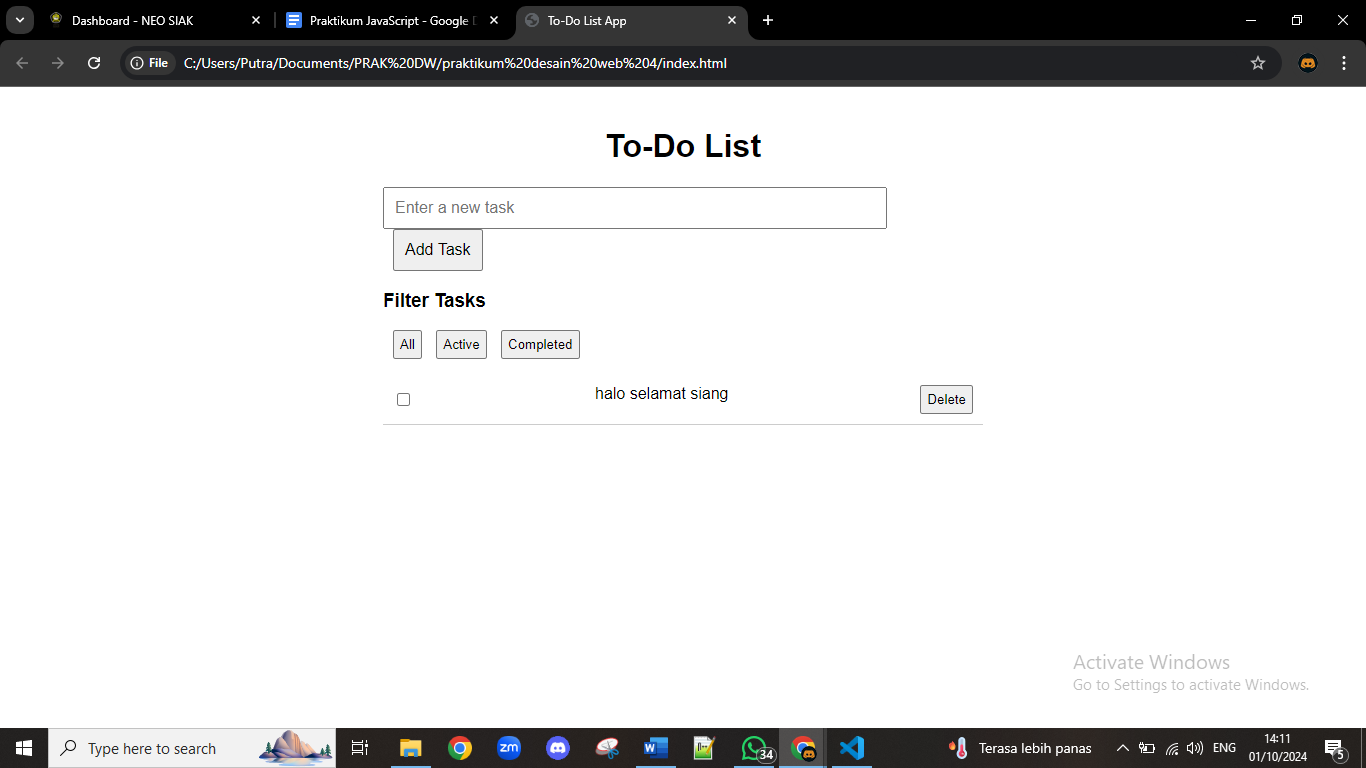
* displayTasks() dan bagian dari filterTasks() mengelola manipulasi DOM. Mereka membersihkan (innerHTML = '') elemen <ul> (daftar tugas), lalu membuat elemen <li> untuk setiap tugas dalam array tasks. Setiap elemen <li> berisi:
* Checkbox yang mencerminkan status selesai.
* Nama tugas sebagai teks.
* Tombol 'Delete' untuk menghapus tugas.
* Semua elemen ini di-append ke dalam <ul> untuk menampilkan daftar tugas di halaman web.

**Penjelasan event handling yang digunakan, seperti pada tombol “Add Task” dan checkbox.**

* **Tombol “Add Task”**: Event listener dipasang pada elemen tombol #add-task-btn, di mana fungsi addTask() dipanggil saat tombol diklik. Ini memungkinkan pengguna menambahkan tugas baru ke daftar.
* **Checkbox**: Setiap tugas memiliki checkbox yang mencerminkan status selesai. Checkbox ini memiliki event listener yang memanggil fungsi toggleTask() ketika diklik, untuk mengubah status completed dari tugas yang bersangkutan.
* **Tombol 'Delete'**: Setiap tugas juga memiliki tombol hapus. Ketika diklik, event listener memanggil fungsi deleteTask() untuk menghapus tugas tersebut.

**BAB IV**

**4.1 HASIL UJICOBA**



**1. Fitur Penambahan Tugas (addTask)**

* **Pengujian**: Saat pengguna memasukkan nama tugas di input dan mengklik tombol **"Add Task"**, tugas ditambahkan ke dalam daftar.
* **Hasil**: Tugas berhasil ditambahkan, dan muncul di daftar tugas di layar. Jika input kosong, muncul alert dengan pesan "Task cannot be empty!", mencegah tugas kosong ditambahkan.

**2. Fitur Menandai Tugas Selesai (toggleTask)**

* **Pengujian**: Setiap tugas yang ditampilkan memiliki checkbox. Ketika checkbox diklik, status tugas berubah dari "belum selesai" ke "selesai", dan sebaliknya.
* **Hasil**: Checkbox bekerja dengan benar, mengubah status completed dari false menjadi true, serta sebaliknya. Tugas yang selesai diberi tampilan yang berbeda (misalnya, melalui class 'completed').

**3. Fitur Menghapus Tugas (deleteTask)**

* **Pengujian**: Setiap tugas memiliki tombol **"Delete"**. Saat tombol tersebut diklik, tugas tersebut dihapus dari daftar.
* **Hasil**: Tugas berhasil dihapus dari daftar, dan tampilan diperbarui secara langsung tanpa perlu memuat ulang halaman. Tugas yang sudah dihapus tidak lagi muncul di daftar.

**4. Fitur Filter Tugas (filterTasks)**

* **Pengujian**: Fitur ini memungkinkan pengguna memfilter tugas berdasarkan statusnya:
  + **Active**: Menampilkan hanya tugas yang belum selesai.
  + **Completed**: Menampilkan hanya tugas yang sudah selesai.
* **Hasil**:
  + Ketika filter diatur ke **"Active"**, hanya tugas yang belum diselesaikan muncul di daftar, dan tugas yang sudah diselesaikan disembunyikan.
  + Ketika filter diatur ke **"Completed"**, hanya tugas yang selesai muncul, sementara tugas aktif disembunyikan.
  + Tampilan berubah secara dinamis tanpa perlu memuat ulang halaman.

**5. Validasi Input Kosong**

* **Pengujian**: Memasukkan input kosong dan mencoba menambahkan tugas.
* **Hasil**: Aplikasi memunculkan alert dengan pesan "Task cannot be empty!" dan mencegah tugas kosong masuk ke daftar, sesuai harapan.

**6. Tampilan Daftar Tugas (displayTasks)**

* **Pengujian**: Tugas yang ditambahkan atau diperbarui langsung ditampilkan dalam elemen <ul>. Setiap tugas ditampilkan sebagai item dengan checkbox dan tombol delete.
* **Hasil**: Tampilan daftar tugas bekerja dengan baik. Setiap perubahan, seperti penambahan tugas, penghapusan, atau perubahan status (selesai atau belum selesai), diperbarui secara langsung di halaman tanpa memuat ulang.

**4.2 KESIMPULAN**

Aplikasi ini membuat mengelola daftar tugas menjadi mudah dan interaktif. Tugas dapat ditambahkan, ditandai sebagai selesai, dihapus, dan difilter berdasarkan statusnya (aktif atau selesai). Aplikasi ini memiliki fungsi dasar yang diperlukan untuk memantau dan mengorganisir aktivitas harian secara efektif, seperti addTask(), toggleTask(), deleteTask(), dan filterTasks. Aplikasi ini dapat berjalan secara dinamis di sisi klien berkat JavaScript, yang memungkinkan Anda melihat perubahan pada daftar tugas langsung tanpa memuat ulang halaman.

Link github :

<https://github.com/ut4h3re/Prak-DW/tree/main/praktikum%20desain%20web%204>